openpgp
Internet-Draft
Intended status: Standards Track
Expires: 7 May 2026

D. Shaw
Jabberwocky Tech
A. Gallagher, Ed.
PGPKeys.EU
D. Huigens
Proton AG

3 November 2025

OpenPGP HTTP Keyserver Protocol
draft-gallagher-openpgp-hkp-09

Abstract

This document specifies a series of conventions to implement an OpenPGP keyserver using the Hypertext Transfer Protocol (HTTP). As this document is a codification and extension of a protocol that is already in wide use, strict attention is paid to backward compatibility with these existing implementations.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://andrewgdotcom.gitlab.io/draft-gallagher-openpgp-hkp. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-gallagher-openpgp-hkp/.

Discussion of this document takes place on the OpenPGP Working Group mailing list (mailto:openpgp@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/openpgp/. Subscribe at https://www.ietf.org/mailman/listinfo/openpgp/.

Source for this draft and an issue tracker can be found at https://gitlab.com/andrewgdotcom/draft-gallagher-openpgp-hkp.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

This Internet-Draft will expire on 7 May 2026.

Copyright Notice

Table of Contents

1. Introduction

For ease of use, public key cryptography requires a key distribution system. For many years, the most commonly used system has been a keyserver - a server that stores public keys and/or certificates, with a searchable interface. The HTTP Keyserver Protocol is a OpenPGP keyserver implemented using HTTP.

2. Conventions and Definitions

The term "OpenPGP Certificate" is used in this document interchangeably with "OpenPGP Transferable Public Key", as defined in Section 10.1 of [RFC9580].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Keyserver Use Cases

A keyserver is typically used for the following (non-exhaustive) use cases:

3.1. Certificate Discovery

When initiating secure communication with a new correspondent, a client will typically attempt to discover the encryption key(s) that it should use. This is a subtle issue with many security considerations, however many discovery methods involve looking up a certificate on a server using a human-readable identifier such as an email address.

3.2. Certificate Refresh

Certificates in OpenPGP are dynamic objects, therefore it is important to refresh known certificates in order to pick up the latest changes. These changes can include new subkeys and User IDs, updated self-signatures and third-party certifications, and revocations. In some cases it may no longer be possible to search by User ID, therefore it is RECOMMENDED that clients refresh known certificates by primary key fingerprint search.

3.3. Reference Resolution

The OpenPGP wire format includes fields that reference primary keys or subkeys by either Key ID or fingerprint. A client may therefore wish to search for previously unknown certificates based on such a reference.

4. HKP and HTTP

As HKP is implemented over HTTP, everything in [RFC1945] applies to HKP as well, and HKP error codes are the same as the ones used in HTTP.

Due the very large deployment of HKP clients based on HTTP version 1.0, HKP keyservers MUST support HTTP 1.0. HKP keyservers MAY additionally support other HTTP versions.

((dshaw : I expect this to be controversial, but we’ve got tons of deployed code that only works with 1.0. I’d be willing to discuss removing this MUST or make it a SHOULD and add a "implementation notes" section pointing out the problem instead. See issue #5.))

When used over HTTPS, HKP is commonly referred to as "HKPS".

HKP(S) are distinguished from generic use of HTTP(S) by using the URI schemes "hkp" and "hkps" [RFC7595]. HKP is assigned port number 11371 and HKPS is assigned 11372 (although this is rarely used in practice). For reasons of maximum compatibility with firewalls and filtering HTTP proxies, HKP(S) are often served over the standard HTTP(S) port(s) (TCP ports 80 and 443).

By convention and history, HKP defaults to HTTP on TCP port 11371, and HKPS defaults to HTTPS on TCP port 443.

((andrewg : if we assign hkps, we appear to be required to specify a dedicated port, even though nobody uses it. See issue #14.))

A keyserver SHOULD support both HKP and HKPS. A client SHOULD use HKPS, or a transport method with equivalent security properties, such as Tor hidden services [TOR].

4.1. Request Paths

HKP defines three paths, namely "/pks/v2" for the v2 API (Section 5), "/pks/lookup" for legacy lookups (Section 6.1), and "/pks/add" for legacy submission (Section 6.2). Paths beginning with "/pks/v<?>" are reserved for future versions of HKP.

A keyserver MAY support requests to other paths under "/pks", but these are outside the scope of this document. These alternative paths have historically been used to provide human-readable interfaces such as HTML forms, and functionality extensions such as [SKS].

4.2. HTTP Status Codes

When a status or error code needs to be returned by a keyserver, the most appropriate HTTP code from [RFC9110] should be used. It is good practice to return the most specific error code possible: for example, returning 404 ("Not Found") rather than 400 ("Bad Request") when a certificate is not found.

This document gives suggested HTTP error codes for several common situations. Note that these are only suggestions, and implementations may have good reasons (such as not revealing the reason why a request failed) for using other error codes.

Clients SHOULD understand the following codes:

Status Code	Description
200 OK	Request succeeded
403 Forbidden	The requested category/operation is not permitted
404 Not found	The search returned no results, or path not found
410 Gone	Requested data has been permanently deleted, e.g. due to RTBF
422 Unprocessable content	Submission was not well formed
501 Not implemented	The requested category/operation is not supported

Table 1: Status Codes

In addition, a client SHOULD understand 3xx redirect codes.

5. The v2 API

The v2 API is a RESTful interface that uses the GET, PUT, POST, DELETE, HEAD, and OPTIONS methods. Specifically, the abs_path (Section 3.2 of [RFC1945]) is built up of the base path "/pks/v2", followed by request-specific URL path components.

5.1. v2 Lookup Format

Certificate lookups are done via an HTTP GET request.

v2 lookups normally include both "category" and "identifier" URL path components. They are appended to "/pks/v2" as follows:

GET /pks/v2/<category>/<identifier>

The "category" and "identifier" components MUST be supplied.

When the v2 lookup format is being used, v2 output format (Section 7.1) MUST be returned.

The v2 lookup format is designed so that a basic HKP service can be implemented using static files.

Category	Identifier format	Output data	See
certs/by-identity	identity	Certificate bundle	Section 5.1.1
certs/by-vfingerprint	versioned fingerprint	Certificate bundle	Section 5.1.2
certs/by-keyid	key ID	Certificate bundle	Section 5.1.3
canonical	identity	Canonical bundle	Section 5.1.4
index	identity	Index of certificates	Section 5.1.5
prefixlog	date	List of prefixes	Section 5.1.6

Table 2: v2 Lookup Categories

5.1.1. The "certs/by-identity" Lookup Category

A keyserver MAY support the "certs/by-identity" lookup category.

GET /pks/v2/certs/by-identity/<identifier>

The "certs/by-identity" category identifies each certificate by matching the contents of its User ID packet(s) (Section 5.1.9). The response to a successful "certs/by-identity" request is a certificate bundle as specified in Section 9, which MUST NOT be ASCII-armored.

A keyserver SHOULD limit the returned certificate bundle to a reasonable length. Results SHOULD be sorted in order of decreasing confidence in the identity link (Section 8), and then by creation date (most recent first). A keyserver MAY choose to only return results where the identity being searched for exceeds a minimum confidence value, and MAY treat "certs/by-identity" as a synonym for "canonical". If no certificates match the request, the keyserver SHOULD return an appropriate HTTP error code such as 404 ("Not Found").

5.1.2. The "certs/by-vfingerprint" Lookup Category

A keyserver MAY support the "certs/by-vfingerprint" lookup category.

GET /pks/v2/certs/by-vfingerprint/<identifier>

The "certs/by-vfingerprint" category identifies each certificate by the versioned fingerprint of its primary key or a subkey. The versioned fingerprint is provided in the "identifier" path component in hexadecimal encoding, without a preceding "0x". The hexadecimal digits are not case sensitive.

A versioned fingerprint consists of one octet of fingerprint version number and N octets of fingerprint. This is the same octet sequence used in the Issuer Fingerprint and Intended Recipient Fingerprint subpackets (Section 5.2.3 of [RFC9580]).

A keyserver:

* SHOULD include matches with both primary key and subkey fingerprints.

* MAY omit matches with encryption-only subkeys.

If no certificates match the request, the keyserver SHOULD return an appropriate HTTP error code such as 404 ("Not Found").

5.1.3. The "certs/by-keyid" Lookup Category

A keyserver MAY support the "certs/by-keyid" lookup category.

GET /pks/v2/certs/by-keyid/<identifier>

The "certs/by-keyid" category identifies each certificate by the Key ID of its primary key or a subkey (Section 5.5.4 of [RFC9580]). The Key ID is provided in the "identifier" path component as 16 hexadecimal digits, without a preceding "0x". The hexadecimal digits are not case sensitive.

A keyserver:

* SHOULD include matches with both primary key and subkey Key IDs.

* MAY omit matches with encryption-only subkeys.

If no certificates match the request, the keyserver SHOULD return an appropriate HTTP error code such as 404 ("Not Found").

"certs/by-keyid" is only required for locating a signing key that made either a V3 signature, or a V4 signature with an Issuer Key ID subpacket and no Issuer Fingerprint subpacket. Issuer Key ID subpackets are not specified for use in later signature versions (Section 5.2.3.12 of [RFC9580]), and so certificates with versions greater than 4 MUST NOT be returned in response to a "certs/by-keyid" request.

5.1.4. The "canonical" Lookup Category

A keyserver MAY support the "canonical" lookup category.

GET /pks/v2/canonical/<identifier>

The "canonical" category is similar to the "certs/by-identity" category, but is intended specifically for certificate discovery (Section 3.1). A keyserver MUST return either the canonical bundle of the identity being searched for (Section 5.2.7), or a 404 Not Found error.

5.1.5. The "index" Lookup Category

A keyserver MAY support the "index" lookup category.

GET /pks/v2/index/<identifier>

This requests a list of certificates on the keyserver whose User IDs match the identity given in the "identifier" path component (Section 5.1.9). This list is returned in JSON format as specified in Section 7.1.1.

A keyserver SHOULD limit the returned index to a reasonable length. Results SHOULD be sorted in order of decreasing confidence in that identity (Section 8), and then by creation date (most recent first). A keyserver MAY choose to only return results where the identity being searched for exceeds a minimum confidence value. If no certificates match the request, the keyserver SHOULD return an appropriate HTTP error code such as 404 ("Not Found").

5.1.6. The "prefixlog" Lookup Category

A keyserver MAY support the "prefixlog" lookup category.

GET /pks/v2/prefixlog/<identifier>

"prefixlog" requests a list of fingerprint prefixes that indicate which certificates have been modified since 00:00:00 UTC on a specific date. The date is provided in the "identifier" path component using the "full-date" format as per Section 5.6 of [RFC3339], i.e. "2025-12-31".

The returned data is a list of CRLF-separated, hexadecimal-encoded, primary key fingerprint prefixes, and each prefix is truncated at a hex-digit (nybble) boundary. Fingerprint prefixes do not include the version number. The keyserver SHOULD calibrate the prefix length so that it is long enough to provide collision resistance, but short enough to maintain a useful anonymity cohort. For example, if the prefix length was too short, clients could be induced to make excessive numbers of network requests. A client MUST NOT make any assumptions about the length of the prefixes returned.

A client that wishes to update its local keystore from a keyserver MAY first make a "prefixlog" request with the date of the last successful refresh. It can then compare the returned list of prefixes to see if any of them are present in its local keystore, and make subsequent "certs/by-vfingerprint" requests as appropriate (Section 5.1.2). In this way, it can avoid making unnecessary requests that will return no updates, but will still leak information to the keyserver.

Note that the prefixlog endpoint always returns fingerprint prefixes, regardless of fingerprint version. This contrasts with v4 Key IDs, which are constructed from fingerprint suffixes.

A keyserver MUST NOT support indexing or downloading certificates by prefix.

5.1.7. OPTIONS Lookup Method

A client MAY attempt to detect which lookup categories a keyserver supports by making an OPTIONS request (Section 9.3.7 of [RFC9110]) to a lookup path with the "category" path component present but the "identifier" component absent.

A keyserver that supports the lookup category MAY respond with:

* a 200 "OK" code,

* an "Allow:" response header that includes the value "GET" (Section 10.2.1 of [RFC9110]).

Otherwise, it SHOULD respond with an error code such as 501 "Not Implemented". Note however that a keyserver that does not support OPTIONS (for example, if it is implemented using static files) MAY return another error code such as 404 "Not Found" or 405 "Method Not Allowed".

A client SHOULD NOT attempt to make a GET request to a lookup path without both the "category" and "identifier" path components present. A keyserver SHOULD return an error code such as 403 "Forbidden" to such a request.

5.1.8. HEAD Lookup Method

A client MAY attempt to retrieve the metadata of a certificate or certificate bundle by making a HEAD request (Section 9.3.2 of [RFC9110]) to a full lookup path, with the "category" and "identifier" components present. A keyserver SHOULD respond with the same header fields that it would have responded with if a GET request had been made.

5.1.9. v2 Identity Lookups

The format of User IDs in OpenPGP has historically been vaguely specified and loosely interpreted (see [I-D.dkg-openpgp-userid-conventions]). A particular identity may therefore be represented by an arbitrary number of different User ID packets. Implementers should bear in mind that end users will typically search for identities, and not specific representations of that identity.

For example, the User ID strings Andrew Gallagher <andrew@example.com> and Andrew B. Gallagher (work email) <andrew@example.com> are both representations of the underlying identity andrew@example.com.

v2 "certs/by-identity", "canonical" and "index" lookup requests MUST only return results if the "identifier" path component exactly matches one of the following:

* The portion between angle brackets (<...>) in an email-address style User ID.

* The full text string in a non-email-address User ID.

A keyserver SHOULD parse an email-address style User ID defensively. In particular, if there is more than one substring of a User ID that could reasonably be interpreted as an email address, then a keyserver SHOULD NOT return an identity match on either substring.

Text lookups SHOULD NOT be case sensitive. DNS names are not case sensitive, therefore any identity that contains a DNS name (including email addresses) cannot be reliably located using case sensitive matching. Since the interpretation of User IDs is application-specific, a client MUST check that any User IDs returned from a keyserver are correctly case matched for the intended application.

5.1.10. Lookup Examples

Get all certificates containing the email address "dshaw@example.com":

https://keys.example.com/pks/v2/certs/by-identity/dshaw@example.com

Get certificate 0xCAFEDADACAFEDADACAFEDADACAFEDADACAFEDADACAFEDADACAFEDADACAFEDADA (v6 fingerprint):

https://keys.example.com/pks/v2/certs/by-vfingerprint/06cafedadacafedadacafedadacafedadacafedadacafedadacafedadacafedada

Get certificate 0xDEADBEEFDECAFBAD (64-bit Key ID):

https://keys.example.com/pks/v2/certs/by-keyid/DEADBEEFDECAFBAD

5.2. v2 Submission Format

A keyserver MAY accept submissions via an HTTP POST or PUT request, as specified in Section 8.3 of [RFC1945], and Section 8.2.3 of [RFC1866].

v2 submission requests include a "category" URL path component, and optionally an "identifier" path component. These are appended to "/pks/v2" as follows:

POST /pks/v2/<category>
PUT /pks/v2/<category>/<identifier>

The "category" path component MUST be supplied. In PUT submission requests, the "identifier" path component is required. In POST submission requests, the "identifier" path component is omitted.

Category	Method	Input data	Output data	See
certs	POST	certificate bundle	submission response	Section 5.2.1
sendtoken	POST	email address	(none)	Section 5.2.2
canonical	PUT	canonical bundle	submission response	Section 5.2.3

Table 3: v2 Submission Request Categories

Unless otherwise specified, a keyserver SHOULD respond to v2 submissions with a JSON document as described in Section 7.2.

If a keyserver does not support submission via HTTP, then requests to do so should return an appropriate HTTP error code, such as 403 ("Forbidden") if certificate submission has been disallowed, or 404 ("Not Found") if the server does not support the requested submission category.

5.2.1. The v2 "certs" Submission Category

A keyserver MAY support the "certs" submission category.

POST /pks/v2/certs

A keyserver that implements it SHOULD support the basic submission workflow described in Section 5.2.4. It MAY support other workflows; if so it SHOULD advertise them as described in Section 5.2.6.

A keyserver MAY verify the request and reject any submissions that cannot be verified. This verification SHOULD use a reliable means of authentication, such as login credentials or a Bearer token (Section 5.2.4.1).

5.2.2. The v2 "sendtoken" Submission Category

A keyserver MAY support the "sendtoken" submission category.

POST /pks/v2/sendtoken

The body of the POST request is a single email address. It SHOULD have a content-type of "text/plain". The keyserver SHOULD respond with an empty document.

A keyserver that supports the "sendtoken" request category SHOULD attempt to verify the email address by sending a time-limited Bearer token via email. A client that receives the verification email can then use the token in a canonical submission request (Section 5.2.3). A keyserver MAY limit the number and frequency of verification requests.

The verification email SHOULD contain a multipart/alternative message with two parts:

* A JSON-LD structured mail document [I-D.ietf-sml-structured-email].

* A human-readable document containing instructions for manual submission.

The schema for the JSON-LD document is ((TBC, issue #40)), and it contains the following fields:

Field	Type	Description
email	string	email address being verified
url	string	URL to be used for submission
token	string	verification token
expires	timestamp	expiry time of the token

Table 4: Structured Mail Fields

Timestamps are given in UTC as per Section 5.6 of [RFC3339].

((TBC: this follows a prove-then-submit model, which is the inverse of KOO's current submit-then-prove process. The rationale is that submit-then-prove often silently degrades to "submit-then-forget-to-prove". Failed advance proofs are less likely to be mis-reported as a success. In addition, prove-then-submit more easily generalises to other forms of verification. Is this defensible? issue #41))

5.2.3. The v2 "canonical" Submission Category

A keyserver MAY support the "canonical" submission category.

PUT /pks/v2/canonical/<identifier>

The request path is the same as the one used for the "canonical" lookup category (Section 5.1.4).

A keyserver that implements it SHOULD support the basic submission workflow described in Section 5.2.4. It MAY support other workflows; if so it SHOULD advertise them as described in Section 5.2.6.

This instructs the server that for the identity (Section 5.1.9) supplied in the request path:

* The certificates contained in the submission that have the corresponding identity are regarded by the owner as canonical for that identity,

* The owner wishes for these certificates to be served in the same order and format that they appear in the submission, and:

* Any certificates for that identity that are not contained in the submission are not (or no longer) canonical for that identity.

A keyserver MUST verify the request and reject any submissions that cannot be verified. This verification SHOULD use a reliable means of authentication, such as login credentials or a Bearer token (Section 5.2.4.1). Once the keyserver verifies the submission, it stores the resulting canonical bundle (Section 5.2.7) and updates any internal confidence values as necessary (Section 8).

5.2.4. Basic Submission

Basic submission uses a content-type of "application/pgp-keys;armor=no" (Section 5 of [I-D.gallagher-openpgp-media-types]). The body of the POST or PUT request contains a certificate bundle as specified in Section 9, which MUST NOT be ASCII-armored.

5.2.4.1. Token Authentication

When using token authentication with a basic submission workflow, the client adds an Authentication request header containing a Bearer token obtained via a "sendtoken" request (Section 5.2.2).

Authentication: Bearer <token>

This token MUST correspond to one of the identities present in the canonical bundle being submitted. A client SHOULD remove any User IDs not related to the token. A keyserver MAY reject a canonical bundle that contains User IDs not related to the token.

5.2.5. Advanced Submission

Advanced submission uses a content-type of "multipart/form-data". The body of the POST or PUT request contains a multipart with one or more parts. The required part has a content-type of "application/pgp-keys;armor=no" (Section 5 of [I-D.gallagher-openpgp-media-types]) and MUST be named "keytext". This part contains a certificate bundle as specified in Section 9, which MUST NOT be ASCII-armored.

Other parts MAY be included as required by an advanced submission workflow. No advanced submission workflows are currently specified.

5.2.6. Submission Feature Detection Using OPTIONS

A client MAY attempt to detect which certificate submission workflows a keyserver supports by making an OPTIONS request (Section 9.3.7 of [RFC9110]) to a submission path with the "category" path component present but the "identifier" path component absent. A keyserver that supports v2 submission SHOULD respond with:

* a 200 code,
* an "Allow:" response header that includes the value "POST" (Section 10.2.1 of [RFC9110]),
* one or more "Accept:" response header(s) (Section 12.5.1 of [RFC9110]).

Accept response media types MAY include:

* application/pgp-keys - basic submission without proof
* application/pgp-keys;proof=tokens - basic submission with token proof
* multipart/form-data;proof=dkim - advanced submission with DKIM proof (EXPERIMENTAL)

((TODO: check the requirements for CORS preflight, issue #38))

5.2.7. Canonical Bundles

A certificate bundle submitted or returned via a "canonical" request is called a "canonical bundle". A canonical bundle represents both the list of certificates that the key holder wishes to be associated with an identity, and their preferred form of those certificates. A keyserver SHOULD serve only canonical bundles in response to a "canonical" lookup request. A keyserver MAY serve full copies of the

matching certificate(s) in response to a "certs" lookup request. For example, a full copy may include valid certificate components obtained by non-canonical submission, but not present in the canonical bundle.

When submitting a canonical bundle, any new certificate components SHOULD be merged into the full copy of the corresponding certificate, if the keyserver supports it. Any valid key or self-certification revocations known to the keyserver SHOULD be merged into the corresponding canonical bundle(s), if any. This applies both to revocations already present in the key store, and to those submitted at any later date. A keyserver SHOULD NOT modify a canonical bundle in any other way.

A keyserver that accepts submissions MUST allow a valid key revocation certificate for any key to be submitted without identity verification. A valid key revocation signature SHOULD be applied to all copies of the key that it revokes, and SHOULD be served in response to all requests for that key. An implementation MAY however omit some revocations for brevity - for example, if two valid revocations exist with different timestamps but otherwise identical effect, an implementation MAY serve the older revocation and omit the newer one.

Other methods such as [I-D.dkg-openpgp-1pa3pc] are more appropriate for controlling the form of certificates returned by non-canonical requests.

5.2.8. Submission Examples

((TODO: issue #39))

6. The Legacy API

For backwards compatibility with the existing installed client base, a Legacy API is defined. New implementations SHOULD use the v2 API.

6.1. Legacy Lookup Format

Legacy certificate lookups are done via an HTTP GET request. Specifically, the abs_path (Section 3.2 of [RFC1945]) is built up of the base path "/pks/lookup", followed by a request-specific query string (Section 8.2.2 of [RFC1866]). No URL path components under "/pks/lookup" are used.

Most Legacy lookups contain both the "op" (operation) and "search" variables. These roughly correspond to the "category" and "identifier" components of the v2 API, but with subtly different

semantics. The "op" variable determines what operation the keyserver will execute, and the "search" variable determines which certificates are operated on.

The "op" and "search" variables are supplied as HTTP query strings, in the form "<variable-name>=<value>":

/pks/lookup?op=<op>&search=<search>[&...]

The "op" variable MUST be supplied, and the "search" variable MUST be supplied unless the "stats" operation is being requested (Section 6.1.6).

There may also be modifier variables, as specified in Section 6.3 below. Keyservers MUST ignore any unknown query string parameters.

6.1.1. The "op" (Operation) Lookup Variable

The "op" (operation) variable specifies the lookup operation to be performed on the keyserver. The "op" variable is generally accompanied by a "search" variable to specify the certificates that should be looked up.

If a particular operation is not supported, the keyserver SHOULD return an appropriate HTTP error code such as 501 ("Not Implemented"). The server SHOULD NOT return an error code (such as 404 ("Not Found")) that could be interpreted by the client as an explicit statement of non-existence.

Operation	Search format	Output data	See
get	text	Certificate bundle	Section 6.1.2
hget	SKS hash	Certificate bundle	Section 6.1.3
index	text	Index of certificates	Section 6.1.4
vindex	text	Index of certificates	Section 6.1.5
stats	(none)	Implementation info	Section 6.1.6

Table 5: Legacy Lookup Operations

6.1.2. The "get" Operation

A keyserver MAY support the "get" operation.

The "get" operation requests certificates from the keyserver by textual search. A string that specifies which certificate(s) to return is provided in the "search" variable.

The response to a successful "get" operation is an ASCII-armored certificate bundle as specified in Section 9.

A keyserver SHOULD limit the returned certificate bundle to a reasonable length. Results from a User ID search SHOULD be sorted in order of decreasing confidence in that identity (Section 8), and then by creation date (most recent first). A keyserver MAY choose to only return results where the identity being searched for has a nonzero confidence value. If no certificates match the request, the keyserver SHOULD return an appropriate HTTP error code such as 404 ("Not Found").

6.1.3. The "hget" (hash get) Operation

A keyserver MAY support the "hget" operation.

"hget" requests a certificate from a keyserver by specifying its [SKS] digest. The digest is provided in the "search" variable in hexadecimal encoding, without a preceding "0x". The hexadecimal digits are not case sensitive.

If no certificates match the request, the keyserver SHOULD return an appropriate HTTP error code such as 404 ("Not Found").

6.1.4. The "index" Operation

A keyserver MAY support the "index" operation.

The "index" operation requests a list of certificates on the keyserver that match the text in the "search" variable. Historically, the "index" operation returned a human-readable HTML document containing links for each found certificate, but this is not required.

A keyserver SHOULD limit the returned index to a reasonable length. Results from a User ID search SHOULD be sorted in order of decreasing confidence in that identity (Section 8), and then by creation date (most recent first). If no certificates match the request, the keyserver SHOULD return an appropriate HTTP error code such as 404 ("Not Found").

6.1.5. The "vindex" (verbose index) Operation (Deprecated)

A keyserver MAY support the "vindex" operation. The "vindex" operation is deprecated. Historically, a "vindex" response was the same as "index" with the addition of showing the signatures on each certificate, but this is not required. A server that supports "vindex" SHOULD treat it as a synonym for "index".

6.1.6. The "stats" (statistics/status) Operation (Deprecated)

A keyserver MAY support the "stats" operation. The "stats" operation is deprecated. It is RECOMMENDED to use a URL outside the standard HKP paths (such as "/pks/stats") instead.

The output of the "stats" operation is implementation-dependent, but may include diagnostic output, configuration state, or other metadata. The "search" variable SHOULD NOT be supplied, and SHOULD be ignored if received.

6.1.7. The "search" Lookup Variable

The "search" variable contains arbitrary text encoded as usual for a HTTP URL. This text may represent a Key ID, or fingerprint, or some text from a User ID on the certificate being sought, depending on the operation.

6.1.7.1. Legacy Key ID and Fingerprint Searches

To search for a certificate by the Key ID or fingerprint of a primary key or subkey, a client SHOULD use a v2 lookup and either the "certs/by-keyid" (Section 5.1.3) or "certs/by-vfingerprint" (Section 5.1.2) lookup category (as appropriate).

If making a Legacy lookup, a client SHOULD use the "get" operation and prefix the "search" string with "0x" to indicate a hexadecimal number. Key ID strings are 16 hexadecimal digits (64 bits). Fingerprint strings are either 32 (version 3), 40 (version 4), or 64 (version 6) hexadecimal digits and do not include the version number. The hexadecimal digits are not case sensitive.

A keyserver:

* SHOULD accept fingerprints and MAY accept 64-bit Key IDs in the Legacy lookup "search" variable.

* SHOULD include matches with both primary key and subkey fingerprints and Key IDs.

* MAY omit matches with encryption-only subkeys.

* MUST NOT return results for 32-bit "short Key ID" searches, as these do not provide sufficient collision resistance.

* MUST NOT return certificates with versions later than 4 for Key ID searches.

* MUST NOT return version 6 (or later) certificates in the results for Legacy machine-readable requests, but MAY do so for Legacy human-readable requests (Section 7.3).

V3 certificates are no longer considered secure, but MAY be distributed for historical reference.

6.1.7.2. Legacy Text Searches

To perform a Legacy search for a certificate by the text of a User ID, a client SHOULD use the "get" or "index" operation (as appropriate) and MUST NOT prefix the "search" string with "0x". A keyserver MUST NOT return version 6 (or later) certificates in the results for Legacy machine-readable requests, but MAY do so for Legacy human-readable requests (Section 7.3.1).

Legacy text searches SHOULD only return results if the "search" variable exactly matches one of the following:

* The full text string of a User ID.

* The portion between angle brackets ("<...>") in an email-address style User ID.

Text searches SHOULD NOT be case sensitive. DNS names are not case sensitive, therefore any User ID that contains a DNS name (including email addresses) cannot be reliably located using case sensitive matching. Since the interpretation of User IDs is application-specific, a client MUST check that any User IDs returned from a keyserver are correctly case matched for the intended application.

A client making a Legacy text search SHOULD set the modifier variable "exact=on" (Section 6.3.4).

To ensure that relevant results are returned, it is RECOMMENDED that implementations limit themselves to a subset of UTF-8 when generating both User IDs and search strings:

* Canonicalise to Unicode Normalization Form C [UNF].

* Do not use punycode [RFC3492].

* Use lowercase in the domain part of email addresses.

* Avoid leading or trailing whitespace.

* Avoid control characters, including format effectors (such as tabs or newlines).

A keyserver MUST NOT return legacy text search results for a search string that begins with "0x". Since the "0x" prefix is generally understood as indicating a fingerprint or key ID, violating this convention could enable a key substitution attack.

6.1.8. Legacy Lookup Examples

Search for all certificates containing the email address "dshaw@example.com", using plaintext HTTP:

```
http://keys.example.com:11371/pks/lookup?search=dshaw@example.com&op=index
```

Get certificate 0xDEADBEEFDECAFBADDEADBEEFDECAFBADDEADBEEFDECAFBAD (v4 fingerprint), using HTTPS:

```
https://keys.example.com/pks/lookup?op=get&search=0xDEADBEEFDECAFBADDEADBEEFDECAFBADDEADB
EEFDECAFBAD
```

Get certificate 0xDEADBEEFDECAFBAD (64-bit Key ID), using HTTPS:

```
https://keys.example.com/pks/lookup?op=get&search=0xDEADBEEFDECAFBAD
```

6.2. Legacy Submission Format

The abs_path (Section 3.2 of [RFC1945]) for certificate submission is always "/pks/add" in the Legacy API.

No URL path components under "/pks/add" are used, and there are no mandatory query strings.

The body of the POST message has a content-type of "application/x-www-form-urlencoded". It contains a "keytext" field whose value is an ASCII-armored certificate bundle as specified in Section 9. The ASCII armored certificate bundle should also be urlencoded as specified in Section 8.2.1 of [RFC1866].

There may also be modifier variables, as specified in Section 6.3 below. Modifiers are passed using HTTP query strings as specified in Section 8.2.2 of [RFC1866]. HTTP query strings MAY be given in any order. Keyservers MUST ignore any unknown query strings.

Note that more than one certificate may be submitted in a single transaction.

If a keyserver does not support adding certificates via HTTP, then requests to do so should return an appropriate HTTP error code, such as 403 ("Forbidden") if certificate submission has been disallowed, or 404 ("Not Found") if the server does not support the legacy submission API.

6.3. Legacy Modifier Variables

These variables are used to modify basic Legacy requests.

Name	Context	Value	See
options	any	list of flags	Section 6.3.1
fingerprint	lookup	"on" or "off"	Section 6.3.2
hash	lookup	"on" or "off"	Section 6.3.3
exact	lookup	"on" or "off"	Section 6.3.4

Table 6: Legacy Variable Names

6.3.1. The "options" Variable

This variable takes one or more option values, separated by commas. These are used to modify the behavior of the keyserver on a per-request basis. Each value indicates a boolean flag, where the presence of the value indicates "true" and the absence "false".

Name	Context	See
nm	submission	Section 6.3.1.1
mr	lookup	Section 6.3.1.2

Table 7: Legacy Option Values

6.3.1.1. The "nm" (No Modification) Option

As keyservers may modify submitted certificates to suit a particular policy, this option is used to inform the keyserver that the submitter would rather have the submission fail completely than have the submitted certificate(s) modified. An example of this would be a keyserver that does not accept User IDs with an email address outside of the local domain. If such a certificate was submitted, the keyserver MAY trim any noncompliant User IDs before accepting the certificate. If this option was set, then such a certificate submission SHOULD fail with an appropriate error code such as 422 (Unprocessable content).

"nm" is meaningful for submissions only.

6.3.1.2. The "mr" (Machine-Readable) Option

The machine-readable option instructs the server that a program (rather than a person) is making a Legacy request, so the output SHOULD be in Legacy machine-readable format. If a v2 request format is being used, this option has no effect. See Section 7.3 for the specific details of Legacy machine-readable output.

An implementation that does not wish to provide a human-readable interface MAY choose to behave as if this option is always present. Implementations SHOULD NOT provide a Legacy interface without supporting machine-readable output.

"mr" is meaningful for Legacy lookups only.

6.3.2. The "fingerprint" Variable

This variable takes one argument: "on" or "off". If present and on, it instructs the server to provide the primary key fingerprint for each certificate in a Legacy "index" or "vindex" operation. This variable has no effect on any other operation. The exact format of the displayed fingerprint, like the "index" and "vindex" operations themselves, is implementation defined in Legacy human-readable output. In Legacy machine-readable indexes, a value of "on" indicates that the "keyid" field SHOULD contain the fingerprint, except for v3 certificates (Section 7.3.1). An implementation SHOULD treat this variable as being "on" for all Legacy machine-readable indexes. An implementation MAY decide to ignore this variable and/or set the default behaviour to "on" for Legacy human-readable indexes.

"fingerprint" is meaningful for Legacy lookups only.

6.3.3. The "hash" Variable

This variable takes one argument: "on" or "off". If present and on, it instructs the server to provide the [SKS] digest of each certificate in an "index" or "vindex" operation in the Legacy human-readable output format. This variable has no effect on any other operation, or on Legacy machine-readable output. The exact format of the displayed digest, like the "index" and "vindex" operations themselves, is implementation defined. An implementation MAY decide to ignore this variable and/or set the default behaviour to "on".

"hash" is meaningful for Legacy lookups only.

6.3.4. The "exact" Variable

This variable takes one argument: "on" or "off". If set to "off", it instructs the server that it MAY use non-exact matching for the contents of the "search" variable in text searches (Section 6.1.7.2). How a keyserver handles non-exact text searches is implementation defined. For example, a keyserver MAY use case-insensitive or tokenized searching.

A keyserver implementation SHOULD set the default behaviour to "on" and MAY ignore this variable.

"exact" is meaningful for Legacy lookups only.

7. Output Formats

HKP was originally intended for both human and programmatic use. In general, the Legacy human-readable output is implementation specific. The "machine-readable" option is used to tailor the output of Legacy requests for automated use. For interoperability, the Legacy machine-readable output MUST carefully follow the guidelines given here. A client implementation SHOULD NOT attempt to parse Legacy human-readable output.

The v2 API always returns either non-armored certificate bundles or JSON [RFC8259], depending on the request.

7.1. v2 Output Format

Clients making v2 requests:

* MUST silently ignore any primary keys with unknown versions or algorithms.

* MUST silently ignore any unknown fields in JSON responses.

In response to a v2 request, a keyserver:

* MUST set the HTTP header "Access-Control-Allow-Origin: *", as specified in [CORS].

* MUST use the format specified in Section 7.1.1 when responding to "index" lookups.

* MUST use the format specified in Section 7.2 when responding to "certs" submissions.

* MUST return non-armored (binary) certificate bundles in response to lookup requests.

* MUST set "Content-Type: application/json" for JSON responses.

* MUST set "Content-Type: application/pgp-keys;armor=no" when returning non-armored certificate bundles (Section 5 of [I-D.gallagher-openpgp-media-types]).

* MAY set "Last-Modified:" to indicate the modification date of the requested certificate or certificate bundle (Section 8.8.2 of [RFC9110]).

7.1.1. v2 Indexes

A v2 "index" request SHOULD return a JSON list of certificates. If the search was for an identity, it SHOULD be sorted in decreasing order of confidence (Section 8). Each certificate object contains some or all of the following fields:

Field	Type	Description
version	integer	version of the primary key (REQUIRED)
fingerprint	string	fingerprint of the primary key (REQUIRED)
creation	string	creation date of the key
expiration	string	expiration date of the key
isExpired	boolean	
isRevoked	boolean	
algorithm	algorithm	(Table 9)
userIDs	userID array	(Table 10)
subkeys	subkey array	(Table 11)

Table 8: v2 Index Fields

Field	Type	Description
code	integer	algorithm ID (REQUIRED)
name	string	a human-readable identifier for the algorithm
bitLength	integer	key length in bits (DSA/RSA/ElGamal keys only)

Table 9: v2 Index Algorithm Fields

Field	Type	Description
uidString	string	User ID string contents (REQUIRED)
creation	string	creation date of (the first signature over) the User ID
expiration	string	expiration date of the User ID
isExpired	boolean	
isRevoked	boolean	
confidence	integer	(Section 8)

Table 10: v2 Index UserID Fields

Field	Type	Description
version	integer	version of the subkey (REQUIRED)
fingerprint	string	fingerprint of the subkey (REQUIRED)
creation	string	creation date of the subkey
expiration	string	expiration date of the subkey
isExpired	boolean	
isRevoked	boolean	
algorithm	algorithm	(Table 9)

Table 11: v2 Index Subkey Fields

Fingerprints are given in hexadecimal notation, without any "0x" prefix. Timestamps are given in UTC as per Section 5.6 of [RFC3339]. Algorithm IDs are as specified in Section 9.1 of [RFC9580].

The only required fields are the version and fingerprint of any key material, and the uidString of any User IDs. Implementations MAY omit algorithms, subkeys and User IDs from indexes; however if they are present they MUST contain the required fields.

7.2. v2 and Legacy Submission Responses

A v2 or Legacy submission MAY return an empty response, or it MAY return a JSON object summarising the changes. The JSON object MAY contain any or all of the following fields, each of which is an array of certificate objects:

Field	Type	Description
inserted	certificate array	newly added certificates
updated	certificate array	updated certificates
deleted	certificate array	deleted certificates
ignored	certificate array	certificates with no new information
invalid	certificate array	certificates that could not be processed

Table 12: Submission Response Fields

Each certificate object MUST contain "version" and "fingerprint" fields, as in Table 8.

Certificates in the "ignored" and "invalid" arrays MAY also contain a "comment" field describing the reason for their rejection. The comment is a human-readable string, and MAY be displayed to the user.

7.3. Legacy machine-readable Output

Clients requesting machine-readable output in Legacy lookup requests:

* SHOULD supply "options=mr" (Section 6.3.1.2).
* MUST silently ignore any content preceding or following a returned armored key block.
* MUST silently ignore any primary keys with unknown versions or algorithms.

Keyservers returning Legacy machine-readable output:

* MUST set the HTTP header "Access-Control-Allow-Origin: *", as specified in [CORS].

* MUST return ASCII-armored certificate bundles.

* MUST NOT return version 6 (or later) certificates.

* MUST set "Content-Type: application/pgp-keys" when returning ASCII-armored certificate bundles (the "get" and "hget" operations), as specified in Section 7 of [RFC3156].

* MAY set "Last-Modified:" to indicate the modification date of the requested certificate or certificate bundle (Section 8.8.2 of [RFC9110]).

* MUST use the format specified in Section 7.3.1 when returning indexes (the "index" and "vindex" operations).

* MAY return statistics in JSON format [RFC8259], the schema of which is otherwise implementation-dependent.

ASCII-armored responses MAY be wrapped in any HTML or other text desired, except that the actual certificate data consisting of an initial line break, the "-----BEGIN PGP PUBLIC KEY BLOCK-----" header, the armored certificate data itself, the "-----END PGP PUBLIC KEY BLOCK-----" footer, and a final line break MUST NOT be modified from the form specified in [RFC9580].

7.3.1. Legacy machine-readable Indexes

The Legacy machine-readable index format is a list of newline-separated records, consisting of colon-separated fields. The document is 7-bit clean, and as such is sent with no encoding and Content-Type: text/plain.

The machine-readable response MAY be prefixed by an information record:

info:<version>:<count>

Field	Description
version	the version of this output format
count	the number of certificates returned

Table 13: Legacy Information Record Fields

If this line is not included, or the version information is not supplied, the version number is assumed to be 1. Currently, only version 1 is defined.

Note that "count" is the number of certificates, and not the number of lines returned. That is, it SHOULD match the number of "pub:" lines returned.

The certificate listings themselves are made up of several records per certificate. The first record specifies the primary key:

pub:<keyID>:<code>:<bitLength>:<creation>:<expiration>:<flags>

Field	Description
keyID	fingerprint or long Key ID
code	algorithm ID
bitLength	key length in bits
creation	creation date of the key
expiration	expiration date of the key
flags	letter codes to indicate details of the key

Table 14: Legacy Public Key Record Fields

Since it is not possible to calculate the Key ID from a V3 fingerprint, for V3 primary keys the "keyID" field SHOULD contain the 16-digit long Key ID only. Otherwise, a keyserver SHOULD return a fingerprint if available (Section 6.3.2).

Fingerprints and long Key IDs are given in hexadecimal notation, without any "0x" prefix. Timestamps are given in seconds since midnight on 1st January 1970. Algorithm IDs are as specified in Section 9.1 of [RFC9580].

Following the "pub" record are one or more "uid" records to indicate User IDs on the certificate:

uid:<uidString>:<creation>:<expiration>:<flags>

Field	Description
uidString	User ID string contents
creation	creation date of (the first self-signature over) the User ID
expiration	expiration date of the User ID
flags	letter codes to indicate details of the User ID

Table 15: Legacy User ID Record Fields

The User ID string MUST use urlencoding for anything that isn't 7-bit safe as well as for the ":" and "%" characters. Any other characters MAY be urlencoded, as desired.

The information for the "creation", "expiration", and "flags" fields is taken from the User ID self-signature, if any, and applies to the User ID in question, not to the certificate as a whole.

Primary key and User ID records may contain a "flags" field containing a sequence of alphabetical characters, one per flag. Flags MAY be given in any order. The meaning of "disabled" is implementation-specific. Note that individual flags may be unimplemented, so the absence of a given flag does not necessarily mean the absence of the detail.

Flag	Description
r	revoked
d	disabled
e	expired

Table 16: Legacy Record Flags

Note that empty fields are allowed. For example, a primary key with no expiration date would have the "expirationdate" field empty. Also, a keyserver that does not track a particular piece of information may leave that field empty as well. Colons for empty

fields on the end of each line MAY be left off, if desired. All dates are given in the number of seconds since 1970-01-01T00:00:00 UTC.

For backwards compatibility with the installed client base, Legacy machine-readable lookup requests MUST omit version 6 (and later) certificates from the returned indexes.

8. Confidence

Traditionally, keyservers did not perform any checks against uploaded content other than simple parseability. This left them open to abuse such as flooding and third-party signature spam. Most modern keyservers are now able to perform cryptographic validity tests, and automated content moderation is generally possible.

It is RECOMMENDED that a keyserver assigns a "confidence" value to each User ID in its database. The exact definition of "confidence" is implementation-dependent, but MAY include checks such as email verification or third-party certifications.

A keyserver MAY represent confidence as a number between 0 and 255, where values of 120 or greater indicate "complete confidence", in a v2 Index User ID object (Table 10). This is numerically compatible with the "trust amount" field specified in Section 5.2.3.21 of [RFC9580], but it is not required to calculate it in the same way or represent it internally as an integer value. The confidence field in a v2 Index User ID object is intended as a heuristic for sorting and filtering responses to lookup requests, and is not a replacement for cryptographic verification. A client SHOULD NOT rely on this confidence field, and SHOULD perform its own checks. A keyserver that wishes to publish a cryptographically-verifiable statement about its internal confidence value MAY do so using a certification signature.

9. Certificate Bundle Format

HKP uses a "certificate bundle" as its primary data representation for both input and output.

A certificate bundle is a sequence of one or more OpenPGP certificates (Transferable Public Keys), concatenated directly, as specified in Sections 10.1 and 3.6 of [RFC9580]. An ASCII-armored certificate bundle is a certificate bundle that has been encoded as a single armored block, as specified in Section 6.2 of [RFC9580]. The Legacy API uses ASCII-armored certificate bundles exclusively, whereas the v2 API uses non-armored certificate bundles exclusively.

Certificate bundles are often called "keyrings", however the term "keyring" is used to refer to several related but distinct concepts:

* A sequence of one or more Transferable Public Keys ("public keyring")

* A sequence of one or more Transferable Secret Keys ("private/secret keyring")

* A sequence of packets forming a single Transferable Public Key

* A sequence of packets forming a single Transferable Secret Key

It is therefore RECOMMENDED that implementations avoid using the term "keyring" without qualification.

9.1. Detached Revocations

For OpenPGP certificates prior to version 6, revocation signatures have customarily been distributed as a detached "revocation certificate", as per Section 10.1.3 of [RFC9580]. An HKP server SHOULD allow submission of these detached revocations.

An HKP implementation MAY accept or serve an ASCII-armored "mixed certificate bundle" where one or more revoked certificates have been replaced by their detached revocation certificate(s). Such a "mixed certificate bundle" MUST be sorted so that all detached revocation certificates appear first. This ensures that detached revocations cannot be mistaken for signatures over another primary key.

Mixed certificate bundles MUST NOT be served in responses to v2 lookup requests.

10. Certificate Discovery Using HKPS

SRV records [RFC2782] are commonly used by clients to discover the location of internet services. We can use the "openpgpkey" service name to locate an HKPS service for certificate discovery. HKP clients SHOULD support SRV records.

10.1. The "openpgpkey" SRV Record

The service name for HKPS discovery is "openpgpkey", and the protocol name is "https". The service and protocol labels are therefore "_openpgpkey" and "_https" respectively. These are prepended to the domain part of the email addresses for which certificates are being published:

_openpgpkey._https.<domain-part>

A domain owner wishing to publish OpenPGP certificates for their users would create one or more SRV records at this location, each referencing the address and port number of a keyserver that is authoritative for that domain. Each unique domain part for which email addresses are supported will have its own SRV records.

10.2. Discovery Lookup Over HKPS

When making an HKPS discovery request, a client SHOULD use an HTTP GET request to an authoritative keyserver using the "canonical" lookup category, with the email address in the "identifier" component.

Note that discovery is performed using the "canonical" lookup category, to ensure that the discovery results have been filtered. The domain owner SHOULD take reasonable steps to ensure that any certificates returned from this lookup request are valid certificates for the identity in the "identifier" component. Discovery MAY be implemented at low cost by serving such requests from a static filesystem.

It is RECOMMENDED that certificates returned from discovery are subject to further verification on the client side wherever possible, to mitigate against compromise of the discovery server.

10.3. Certificate Discovery Example

A client trying to locate a certificate for isabella@example.com makes a DNS request for the SRV record at _openpgpkey._https.example.com.. This would return:

```
_openpgpkey._https.example.com. 3600 IN SRV 1 1 443 keyserver.example.com
```

On finding this record, the client makes an HTTP GET request for the following URL:

```
hkps://keyserver.example.com:443/pks/v2/canonical/isabella@example.com
```

The keyserver returns the canonical bundle for isabella@example.com, and the client proceeds to check its validity according to the local policy.

11. Security Considerations

As described here, a keyserver is a searchable database of OpenPGP Certificates accessed over the network. While there may be security considerations arising from distributing arbitrary certificates in this manner, this does not impact the security of OpenPGP itself.

Without some sort of trust relationship between the client and server, information returned from a keyserver in arbitrary search results cannot be trusted by the client until the OpenPGP client actually retrieves and checks the certificate for itself. This is important and must be stressed: without a specific reason to treat information otherwise, all search results SHOULD be regarded as untrustworthy and informational only.

12. IANA Considerations

This document allocates the ports 11371 and 11372, the service names "hkps" and "openpgpkey", and the URI schemes "hkp" and "hkps".

12.1. Updated Registry Entries

IANA is requested to update the contact details for port 11371 in the "Service Name and Transport Protocol Port Number" registry to "Daphne Shaw".

12.2. New Registry Entries

IANA is requested to add the following entries to the "Service Name and Transport Protocol Port Number" registry as per [RFC6335]:

Service Name	Port	Transport Protocol	Description	Contact	Reference
hkps	11372	tcp and udp	OpenPGP HTTP Keyserver (Secure)	Daphne Shaw	This document
openpgpkey			OpenPGP Certificate Discovery	Daphne Shaw	This document

Table 17: Service Name and Transport Protocol Port Number Registry

IANA is requested to add the following entries to the "URI Schemes" registry as per [RFC7595]:

URI Scheme	Description	Status	Reference
hkp	OpenPGP HTTP Keyserver	Permanent	This document
hkps	OpenPGP HTTP Keyserver (Secure)	Permanent	This document

Table 18: Uniform Resource Identifier (URI) Schemes Registry

13. References

13.1. Normative References

[CORS] "Cross Origin Resource Sharing", n.d., <https://fetch.spec.whatwg.org/#cors-protocol>.

[I-D.gallagher-openpgp-media-types] Gallagher, A., "Media Types for OpenPGP", Work in Progress, Internet-Draft, draft-gallagher-openpgp-media-types-00, 8 August 2025, <https://datatracker.ietf.org/doc/html/draft-gallagher-openpgp-media-types-00>.

[I-D.ietf-sml-structured-email] Happel, H., "Structured Email", Work in Progress, Internet-Draft, draft-ietf-sml-structured-email-05, 21 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-sml-structured-email-05>.

[RFC1866] Berners-Lee, T. and D. Connolly, "Hypertext Markup Language - 2.0", RFC 1866, DOI 10.17487/RFC1866, November 1995, <https://www.rfc-editor.org/rfc/rfc1866>.

[RFC1945] Berners-Lee, T., Fielding, R., and H. Frystyk, "Hypertext Transfer Protocol -- HTTP/1.0", RFC 1945, DOI 10.17487/RFC1945, May 1996, <https://www.rfc-editor.org/rfc/rfc1945>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000, <https://www.rfc-editor.org/rfc/rfc2782>.

[RFC3156] Elkins, M., Del Torto, D., Levien, R., and T. Roessler, "MIME Security with OpenPGP", RFC 3156, DOI 10.17487/RFC3156, August 2001, <https://www.rfc-editor.org/rfc/rfc3156>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/rfc/rfc3339>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/rfc/rfc8259>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/rfc/rfc9110>.

[RFC9580] Wouters, P., Ed., Huigens, D., Winter, J., and Y. Niibe, "OpenPGP", RFC 9580, DOI 10.17487/RFC9580, July 2024, <https://www.rfc-editor.org/rfc/rfc9580>.

[UNF] Whistler, K., "Unicode Normalization Forms", n.d., <https://www.unicode.org/reports/tr15/>.

13.2. Informative References

[I-D.dkg-openpgp-1pa3pc] Gillmor, D. K., "First-Party Approved Third-Party Certifications in OpenPGP", Work in Progress, Internet-Draft, draft-dkg-openpgp-1pa3pc-02, 6 September 2024, <https://datatracker.ietf.org/doc/html/draft-dkg-openpgp-1pa3pc-02>.

[I-D.dkg-openpgp-userid-conventions]
Gillmor, D. K., "OpenPGP User ID Conventions", Work in Progress, Internet-Draft, draft-dkg-openpgp-userid-conventions-00, 25 August 2023, <https://datatracker.ietf.org/doc/html/draft-dkg-openpgp-userid-conventions-00>.

[RFC3492] Costello, A., "Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)", RFC 3492, DOI 10.17487/RFC3492, March 2003, <https://www.rfc-editor.org/rfc/rfc3492>.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <https://www.rfc-editor.org/rfc/rfc6335>.

[RFC7595] Thaler, D., Ed., Hansen, T., and T. Hardie, "Guidelines and Registration Procedures for URI Schemes", BCP 35, RFC 7595, DOI 10.17487/RFC7595, June 2015, <https://www.rfc-editor.org/rfc/rfc7595>.

[SKS] "Synchronising Key Server Wiki", n.d., <https://github.com/sks-keyserver/sks-keyserver/wiki>.

[TOR] "Tor Specifications", n.d., <https://spec.torproject.org/>.

Appendix A. Acknowledgments

This document is a formalization and extension of HKP, originally implemented in the PKS keyserver by Marc Horowitz, which in turn was based on earlier work by Brian LaMacchia and Michael Graff. The "prefixlog" request category is based on an earlier proposal by Daniel Kahn Gillmor and Vincent Breitmoser.

The authors would like to thank Peter Gutmann for his work on the Certstore protocol, some of which was applicable here, and the members of the pgp-keyserver-folk mailing list who contributed valuable comments and suggestions. They would also like to thank Bart Butler, Daniel Kahn Gillmor, Heiko Sch辰fer, Justus Winter and Vincent Breitmoser for help with the v2 request format.

Appendix B. Document History

Note to RFC Editor: this section should be removed before publication.

B.1. Changes Between draft-gallagher-openpgp-hkp-08 and draft-gallagher-openpgp-hkp-09

* Refactored to separate v2 and Legacy sections.
* v2 API is now RESTful.
* v2 uses "category" and "identifier" instead of "op" and "search".
* v2 keywords are less terse.
* Defined "canonical bundle".
* Defined "identity".
* Resurrected SRV-based discovery.
* Added basic vs advanced submission.
* Added structured email.
* Specified bearer token authentication.
* Specified use of OPTIONS and HEAD.
* Specified the Last-Modified response header.
* Clarify subkey lookups.
* Sideline inexact matching.
* Discourage badly-behaved User IDs.
* Removed some misused HTTP return codes.
* Added tabular summaries.
* Added Daniel Huigens as co-author.

B.2. Changes Between draft-gallagher-openpgp-hkp-07 and draft-gallagher-openpgp-hkp-08

* Verified uploads now use prove-then-submit workflow.
* Removed SRV and discovery discussion (temporarily?).
* Added definitions of "discovery", "refresh" and "reference resolution".
* Normalised "certificate" terminology and warned about unqualified use of "keyring".
* Renumbered HKPv1 to HKPv2 for avoidance of confusion, and reordered path components.
* HKPv2 is now explicitly a binary protocol, and uses simplified paths.
* Defined v2 submission requests and "cb" option.
* Explicitly forbade mixed certificate bundle responses to v2 lookups.
* "since" is now "prefixlog".
* "get" operation is now MAY.
* "x-*" parameters are no longer specified (as per RFC6648).
* Fixed several errata.
* Expanded commentary and guidance.

B.3. Changes Between draft-gallagher-openpgp-hkp-06 and draft-gallagher-openpgp-hkp-07

* Added "authget" and "since" operations.
* Defined confidence.
* Added more explicit guidance re sorting, filtering and error codes.
* Key version MR output field is now "keyversion" to distinguish from the output format version.

B.4. Changes Between draft-gallagher-openpgp-hkp-05 and draft-gallagher-openpgp-hkp-06

* Updated references.

B.5. Changes Between draft-gallagher-openpgp-hkp-04 and draft-gallagher-openpgp-hkp-05

* Allow detached revocations in keyrings.
* Redesigned v2 request format to use path components for required fields.
* Added openpgpkey discovery file.
* Added "vfpget" and "kidget" operations.
* Added meaningful "exact" semantics.
* HKPS is now SHOULD.
* Defined port 11372.
* IANA registry tables.
* Deprecated op=stats.

B.6. Changes Between draft-gallagher-openpgp-hkp-03 and draft-gallagher-openpgp-hkp-04

* Reworded certificate lookups section for clarity.
* Separate section for keyring format.
* Specify detached revocations.
* Updated references.

B.7. Changes Between draft-gallagher-openpgp-hkp-02 and draft-gallagher-openpgp-hkp-03

* Clients SHOULD supply the v=1 api-versioning variable.
* machine-readable output includes key version field.
* Clients MUST silently ignore leading and trailing cruft, trailing unknown fields, and unknown flags.

* Clients MUST silently ignore keys with unknown versions or algorithms.

* All other m-r index specs (CORS, Content-Type etc.) are now MUST.

* Included the hash variable from SKS.

B.8. Changes Between draft-gallagher-openpgp-hkp-01 and draft-gallagher-openpgp-hkp-02

* Tightened up BCP-14 language.

* Included op=hget from SKS.

* Options now strictly boolean with default false, variables less strict.

* More detail about HTTP status code usage.

B.9. Changes Between draft-shaw-openpgp-hkp-00 and draft-gallagher-openpgp-hkp-01

* Improved text structure.

* Added references to HTTPS/HKPS, and hkp:/hkps: URL schemes.

* Forbade short IDs and deprecated V3 keys.

* Included op=stats from SKS.

* Mentioned CORS.

* Made use of terminology more consistent.

* Replaced custom status codes with standard HTTP status codes.

Authors' Addresses

Daphne Shaw
Jabberwocky Tech
Email: dshaw@jabberwocky.com

Andrew Gallagher (editor)
PGPKeys.EU
Email: andrewg@andrewg.com

Daniel Huigens
Proton AG
Email: d.huigens@protonmail.com